7

5

TEXAS HOLD'EM POKER

GAME SIMULATION

LET'S HAVE FUN TODAY!

Are you ready?

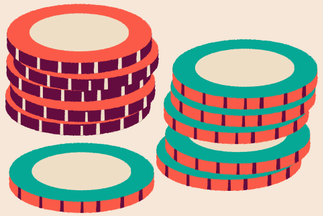
CARD REPRESENTATION (ПРЕДСТАВЛЕННЯ КАРТ)

- Кожна карта в колоді представлена структурою Card, яка має два поля:
- rank (значення карти): від 2 до 13, де 10-13 представляють валета, даму, короля та туза.
- suit (масть): значення масті, наприклад, 0 для піки, 1 для черви, 2 для бубни, 3 для хреста.
- Колода створюється як масив структур Card, який містить усі 52 карти. Використовується функція initialize_deck, яка заповнює масив картами всіх мастей і значень.
- Функція shuffle_deck перемішує колоду, забезпечуючи випадковість у роздачі.

Game Simulation (Симуляція гри)

- Гра симулюється багаторазово за допомогою циклу Монте-Карло (MONTE_CARLO_SIMULATIONS). Це дозволяє розрахувати ймовірність виграшу гравця на різних етапах гри.
- На кожній ітерації симуляції:
- Колода ініціалізується та перемішується.
- Карти гравця і спільні карти видаляються з колоди для уникнення дублювання.
- Роздаються карти опонентам, і на основі отриманих рук визначається переможець.
- Функція calculate_win_probability використовується для обчислення ймовірності виграшу гравця, враховуючи кількість гравців та доступні карти.

Actions (Дії гравця)



- Гравець може виконувати дві основні дії:
- Fold (скинути карти) — коли гравець вирішує не продовжувати гру.
- Continue (продовжити) — гравець погоджується грати далі, розкриваючи наступні спільні карти (флоп, терн, рівер).
- У коді перед кожним етапом запитується дія гравця (scanf(" %c", &action);). Якщо гравець вводить F(Fold), гра закінчується для цього раунду. Якщо вводить C, роздаються нові спільні карти, і обчислюється нова ймовірність виграшу.

Opponent (Суперник)

- Програма дозволяє грати з кількома опонентами (за замовчуванням — один).
- Карти опонентів генеруються випадково з колоди, з якої вже видалені карти гравця і спільні карти.
- Для кожного опонента програма роздає карти та зберігає їх у масиві opponent_hands. Потім функція find_winning_opponent перевіряє, чи є рука опонента, яка перевершує руку гравця з урахуванням спільних карт.

Print Winner (Вивід переможця)

- Після завершення всіх раундів (рівер), програма визначає, хто переміг.
- Функція find_winning_opponent повертає індекс переможця серед опонентів або -1, якщо гравець має найсильнішу комбінацію.
- У кінці гри програма перевіряє результат:
- Якщо гравець має найсильнішу руку, відображається повідомлення, що гравець переміг.
- Якщо перемогу здобув опонент, програма повідомляє, який саме бот виграв раунд.

Функції в коді

initialize_deck()

Створює і ініціалізує колоду карт, заповнюючи її картами з відповідними номіналами та мастями. Це дозволяє отримати повну колоду для гри.

shuffle_deck()

Перемішує колоду карт. Це необхідно для того, щоб кожна нова гра мала випадковий порядок карт, і кожен раунд гри був унікальним.

deal_cards()

Роздає карти як гравцям, так і на спільний борд. Спочатку роздаються карти гравцям, потім — спільні карти (флоп, терн, рівер).

calculate_win_probability()

Використовує симуляцію Монте-Карло для оцінки ймовірності виграшу для гравця на основі поточної ситуації на столі. Це дозволяє гравцю побачити шанси на виграш в залежності від карт і дій.

print_card()

Виводить на екран картки гравця, спільні карти та інформацію про руки, що дозволяє гравцеві слідкувати за ходом гри.

find_winning_opponent()

Оцінює і порівнює руки гравців (гравця і опонентів) для визначення переможця гри на основі покерних комбінацій.

main()

Головна функція, яка контролює основний цикл гри, обробляє введення гравця і викликає інші функції для симуляції гри.

Бібліотеки

stdio.h

Використовується для виведення інформації на екран (функція printf()) та зчитування введення з клавіатури (функція scanf()).

stdlib.h

Використовується для генерації випадкових чисел (функція rand()) і для ініціалізації генератора випадкових чисел (srand()).

time.h

Використовується для ініціалізації генератора випадкових чисел за допомогою поточного часу через функцію time(NULL).

string.h

Може бути використана для маніпулювання рядками, хоча в даному коді її використання мінімальне. (При необхідності можна додавати для додаткових перевірок або маніпуляцій із рядками).

Architectural Decisions (Архітектурні Рішення)

Генерація карт (Deck Generation)

Необхідно мати структуру даних для карт і колоди, а також функцію для їх ініціалізації та перемішування. Можна використовувати масив структур для збереження карт.

Роздача карт (Card Dealing)

Важливо мати механізм, який роздає карти гравцям та спільні карти, при цьому потрібно слідкувати за тим, щоб карти не дублювалися.

Обчислення ймовірності (Win Probability Calculation)

Компонент для оцінки ймовірності виграшу. Це може бути реалізовано за допомогою симуляцій Монте-Карло або аналізу різних можливих результатів на основі поточних карт.

Визначення переможця (Winner Determination)

Необхідно мати алгоритм для порівняння карт гравців з картами на столі, щоб визначити, у кого найкраща покерна комбінація.

Інтерфейс користувача (User Interface)

Взаємодія з користувачем через командний рядок: виведення карт, ймовірностей виграшу та отримання відповідей від гравця (скинути карти або продовжити гру).

THANK YOU FOR YOUR ATTENTION!